(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 186 995 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
13.03.2002 Bulletin 2002/11

(51) Int Cl.[7]: **G06F 9/38**

(21) Application number: **01118582.4**

(22) Date of filing: **27.10.1994**

(84) Designated Contracting States:
DE FR GB IT NL

(30) Priority: **05.11.1993 US 147797**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
99121731.6 / 0 974 894
94116955.9 / 0 652 509

(71) Applicant: **Intergraph Corporation**
Huntsville, Alabama 35824 (US)

(72) Inventor: **Sachs, Howard G.**
Los Altos, CA. 94022-2230 (US)

(74) Representative: **Sparing - Röhl - Henseler**
Patentanwälte
Rethelstrasse 123
40237 Düsseldorf (DE)

Remarks:
This application was filed on 02 - 08 - 2001 as a divisional application to the application mentioned under INID code 62.

(54) Instruction cache associative cross-bar switch

(57) A method for operating a processor, comprising the steps of storing in a memory a plurality of instructions, each instruction being one of a plurality of instruction types, the instructions encoded in frames, each frame including a plurality of instruction slots and template bits that specifies instruction group boundaries within the frame, with an instruction group comprising a set of statically contiguous instructions that are executed concurrently; using a crossbar switch means coupled to a plurality of execution units to issue instructions in the instruction group in parallel to execution units from the plurality of execution units in response to the template bits; wherein each execution unit of the plurality of execution units are one of a plurality of execution unit types; and wherein each instruction type is executed on one or more execution unit types.

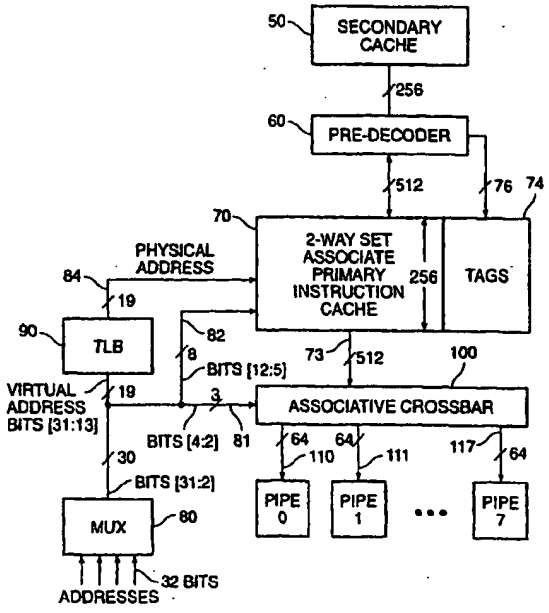

FIG. 2

EP 1 186 995 A1

Description

[0001] This invention relates to a method for operating a processor according to claim 1, a processor according to claim 11 and to a cache memory according to claim 21, and thus to an architecture in which individual instructions may be executed in parallel, as well as to methods and apparatus for accomplishing that.

[0002] A common goal in the design of computer architectures is to increase the speed of execution of a given set of instructions. One approach to increasing instruction execution rates is to issue more than one instruction per clock cycle, in other words, to issue instructions in parallel. This allows the instruction execution rate to exceed the clock rate. Computing systems that issue multiple independent instructions during each clock cycle must solve the problem of routing the individual instructions that are dispatched in parallel to their respective execution units. One mechanism used to achieve this parallel routing of instructions is generally called a "crossbar switch."

[0003] In present state of the art computers, e.g. the Digital Equipment Alpha, the Sun Microsystems SuperSparc, and the Intel Pentium, the crossbar switch is implemented as part of the instruction pipeline. In these machines the crossbar is placed between the instruction decode and instruction execute stages. This is because the conventional approach requires the instructions to be decoded before it is possible to determine the pipeline to which they should be dispatched. Unfortunately, decoding in this manner slows system speed and requires extra surface area on the integrated circuit upon which the processor is formed. These disadvantages are explained further below.

[0004] The invention is defined in claims 1,11 and 21.

[0005] A computing system architecture is provided that enables instructions to be routed to an appropriate pipeline more quickly, at lower power, and with simpler circuitry than previously possible. This invention places the crossbar switch earlier in the pipeline, making it a part of the initial instruction fetch operation. This allows the crossbar to be a part of the cache itself, rather than a stage in the instruction pipeline. It also allows the crossbar to take advantage of circuit design parameters that are typical of regular memory structures rather than random logic. Such advantages include: lower switching voltages (200 - 300 millivolts rather than 3 - 5 volts); more compact design, and higher switching speeds. In addition, if the crossbar is placed in the cache, the need for many sense amplifiers is eliminated, reducing the circuitry required in the system as a whole.

[0006] To implement the crossbar switch, the instructions coming from the cache, or otherwise arriving at the switch, must be tagged or otherwise associated with a pipeline identifier to direct the instructions to the appropriate pipeline for execution. In other words, pipeline dispatch information must be available at the crossbar switch at instruction fetch time, before conventional instruction decode has occurred. There are several ways this capability can be satisfied: In one embodiment this system includes a mechanism that routes each instruction in a set of instructions to be executed in parallel to an appropriate pipeline, as determined by a pipeline tag applied to each instruction during compilation, or placed in a separate identifying instruction that accompanies the original instruction. Alternately the pipeline affiliation can be determined after compilation at the time that instructions are fetched from memory into the cache, using a special predecoder unit.

[0007] Thus, in one implementation, this system includes a register or other means, for example, the memory cells providing for storage of a line in the cache, for holding instructions to be executed in parallel. Each instruction has associated with it a pipeline identifier indicative of the pipeline to which that instruction is to be issued. A crossbar switch is provided which has a first set of connectors coupled to receive the instructions, and a second set of connectors coupled to the processing pipelines to which the instructions are to be dispatched for execution. Means are provided which are responsive to the pipeline identifiers of the individual instructions in the group supplied to the first set of connectors for routing those individual instructions onto appropriate paths of the second set of connectors, thereby supplying each instruction in the group to be executed in parallel to the appropriate pipeline.

[0008] In a preferred embodiment of this invention the associative crossbar is implemented in the instruction cache. By placing the crossbar in the cache all switching is done at low signal levels (approximately 200 - 300 millivolts). Switching at these low levels is substantially faster than switching at higher levels (5 volts) after the sense amplifiers. The lower power also eliminates the need for large driver circuits, and eliminates numerous sense amplifiers. Additionally, by implementing the crossbar in the cache, the layout pitch of the crossbar lines matches the pitch of the layout of the cache.

[0009] A further embodiment of the invention concerns an apparatus in a computing system in which groups of individual instructions are executable in parallel by processing pipelines, said apparatus being used for routing each instruction in a group to be executed in parallel to an appropriate pipeline comprises

a storage for holding at least one group of instructions to be executed in parallel, each instruction in the group having associated therewith a pipeline identifier indicative of the pipeline for executing that instruction;

a crossbar having a first set of connectors coupled to the storage for receiving instructions therefrom and a second set of connectors coupled to the processing pipelines;

means responsive to the pipeline identifier of the individual instructions in the group for routing individual instructions onto appropriate ones of the second

set of connectors, to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

[0010] In this apparatus, the first set of connectors may consist of a set of first communication buses, one for each instruction in the storage;

the second set of connectors may consist of a set of second communication buses, one for each pipeline; and
the means responsive to the pipeline identifier may comprise a set of decoders coupled to the storage to receive as first input signals the pipeline identifiers and in response thereto supply as output signals a switch control signal; and a set of switches, coupled to the decoders, one switch at the intersection of each of the first set of connectors with the second set of connectors, the switches providing connections in response to receiving the switch control signal to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

[0011] A further embodiment of the invention concerns an apparatus in a computing system in which sets of individual instructions are executable in parallel by processing pipelines, said apparatus being used for routing each instruction in a group to be executed in parallel to an appropriate pipeline comprises

a storage for holding a collection of instructions, including at least one set of instructions to be executed in parallel, each instruction in the set having associated therewith a pipeline identifier indicative of the pipeline to which that instruction is to be issued;
a crossbar switch having a first set of connectors coupled to the storage for receiving instructions therefrom and a second set of connectors coupled to the processing pipelines;
selection means connected to receive the set of instructions and connected to receive information about those instructions to be next executed in parallel for supplying in response thereto an output signal indicative of the next set of instructions to be executed in parallel; and
decoder means coupled to receive the output signal and each of the pipeline identifiers of the instructions in the storage for selectively connecting ones of the first set of connectors to ones of the second set of connectors to thereby supply each instruction in the set to be executed in parallel to the appropriate pipeline.

[0012] In this apparatus, the first set of connectors may consist of a set of first communication buses, one for each instruction in the storage;

the second set of connectors may consist of a set of second communication buses, one for each pipeline;
the decoder means may comprise a set of decoders coupled to receive as first input signals the pipeline identifiers and the information about the next group of instructions to be executed by the pipelines and in response thereto supply as output signals a switch control signal; and
the crossbar switch may include a set of switches, one at the intersection of each of the first set of connectors with the second set of connectors, the switches providing connections in response to receiving the switch control signal to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

[0013] Further, the multiplexer may supply an output signal to the decoders to select the next group of instructions to be supplied to the pipelines.

[0014] A further embodiment of the invention concerns a method in a computing system in which groups of individual instructions are executable in parallel by processing pipelines, said method being used for transferring each instruction in a group to be executed through a crossbar switch having a first set of connectors coupled to the storage for receiving instructions therefrom and a second set of connectors coupled to the processing pipelines, said method comprising:

storing in storage at least one group of instructions to be executed in parallel, each instruction in the group having associated therewith a pipeline identifier indicative of the pipeline which will execute that instruction; and
using the pipeline identifiers of the individual instructions in the at least one group of instructions which are to be executed next to control switches between the first set of connectors and the second set of connectors to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

[0015] The step of using may comprise

supplying the pipeline identifiers of the individual instructions in the at least one group of instructions to a corresponding number of decoders, each of which provides an output signal indicative of the pipeline identifiers; and
using the decoder output signals to control the switches between the first set of connectors and the second set of connectors to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

[0016] According to one embodiment, there is provided in a computing system in which groups of individual

instructions are executable in parallel by processing pipelines, a method for supplying each instruction in a group to be executed in parallel to an appropriate pipeline, the method comprising:

storing in storage at least one group of instructions to be executed in parallel, each instruction in the group having associated therewith a pipeline identifier indicative of the pipeline which will execute that instruction; and
using the pipeline identifier of those instructions to be next executed in parallel to control switches in a crossbar switch having a first set of connectors coupled to the storage for receiving instructions therefrom and a second set of connectors coupled to the processing pipelines to thereby supply each instruction in the group to be executed in parallel to the appropriate pipeline.

Figure 1 is a block diagram illustrating a typical environment for a preferred implementation of this invention;
Figure 2 is a diagram illustrating the overall structure of the instruction cache of Figure 1;
Figure 3 is a diagram illustrating one embodiment of the associative crossbar;
Figure 4 is a diagram illustrating another embodiment of the associative crossbar; and
Figure 5 is a diagram illustrating another embodiment of the associative crossbar.

[0017] Figure 1 illustrates the organization of the integrated circuit chips by which the computing system is formed. As depicted, the system includes a first integrated circuit 10 that includes a central processing unit, a floating point unit, and an instruction cache.

[0018] In the preferred embodiment the instruction cache is a 16 kilobyte two-way set-associative 32 byte line cache. A set associative cache is one in which the lines (or blocks) can be placed only in a restricted set of locations. The line is first mapped into a set, but can be placed anywhere within that set. In a two-way set associative cache, two sets, or compartments, are provided, and each line can be placed in one compartment or the other.

[0019] The system also includes a data cache chip 20 that comprises a 32 kilobyte four-way set-associative 32 byte line cache. The third chip 30 of the system includes a predecoder, a cache controller, and a memory controller. The predecoder and instruction cache are explained further below. For the purposes of this invention, the CPU, FPU, data cache, cache controller and memory controller all may be considered of conventional design.

[0020] The communication paths among the chips are illustrated by arrows in Figure 1. As shown, the CPU/FPU and instruction cache chip communicates over a 32 bit wide bus 12 with the predecoder chip 30. The asterisk is used to indicate that these communications are multiplexed so that a 64 bit word is communicated in two cycles. Chip 10 also receives information over 64 bit wide buses 14, 16 from the data cache 20, and supplies information to the data cache 20 over three 32 bit wide buses 18. The predecoder decodes a 32 bit instruction received from the secondary cache into a 64 bit word, and supplies that 64 bit word to the instruction cache on chip 10.

[0021] The cache controller on chip 30 is activated whenever a first level cache miss occurs. Then the cache controller either goes to main memory or to the secondary cache to fetch the needed information. In the preferred embodiment the secondary cache lines are 32 bytes and the cache has an 8 kilobyte page size.

[0022] The data cache chip 20 communicates with the cache controller chip 30 over another 32 bit wide bus. In addition, the cache controller chip 30 communicates over a 64 bit wide bus 32 with the DRAM memory, over a 128 bit wide bus 34 with a secondary cache, and over a 64 bit wide bus 36 to input/output devices.

[0023] As will be described further below, the system shown in Figure 1 includes multiple pipelines able to operate in parallel on separate instructions which are dispatched to these parallel pipelines simultaneously. In one embodiment the parallel instructions have been identified by the compiler and tagged with a pipeline identification tag indicative of the specific pipeline to which that instruction should be dispatched.

[0024] In this system, an arbitrary number of instructions can be executed in parallel. In one embodiment of this system the central processing unit includes eight functional units and is capable of executing eight instructions in parallel. These pipelines are designated using the digits 0 to 7. Also, for this explanation each instruction word is assumed to be 32 bits (4 bytes) long.

[0025] As briefly mentioned above, in the preferred embodiment the pipeline identifiers are associated with individual instructions in a set of instructions during compilation. In the preferred embodiment, this is achieved by compiling the instructions to be executed using a well-known compiler technology. During the compilation, the instructions are checked for data dependencies, dependence upon previous branch instructions, or other conditions that preclude their execution in parallel with other instructions. The result of the compilation is identification of a set or group of instructions which can be executed in parallel. In addition, in the preferred embodiment, the compiler determines the appropriate pipeline for execution of an individual instruction. This determination is essentially a determination of the type of instruction provided. For example, bad instructions will be sent to the bad pipeline, store instructions to the store pipeline, etc. The association of the instruction with the given pipeline can be achieved either by the compiler, or by later examination of the instruction itself, for example, during predecoding.

[0026] Referring again to Figure 1, in normal operation the CPU will execute instructions from the instruc-

tion cache according to well-known principles. On an instruction cache miss, however, a set of instructions containing the instruction missed is transferred from the main memory into the secondary cache and then into the primary instruction cache, or from the secondary cache to the primary instruction cache, where it occupies one line of the instruction cache memory. Because instructions are only executed out of the instruction cache, all instructions ultimately undergo the following procedure.

[0027] At the time a group of instructions is transferred into the instruction cache, the instruction words are predecoded by the predecoder 30. As part of the predecoding process, a multiple bit field prefix is added to each instruction based upon a tag added to the instruction by the compiler. This prefix gives the explicit pipe number of the pipeline to which that instruction will be routed. Thus, at the time an instruction is supplied from the predecoder to the instruction cache, each instruction will have a pipeline identifier.

[0028] It may be desirable to implement the system of this invention on computer systems that already are in existence and therefore have instruction structures that have already been defined without available blank fields for the pipeline information. In this case, in another embodiment of this invention, the pipeline identifier information is supplied on a different clock cycle, then combined with the instructions in the cache or placed in a separate smaller cache. Such an approach can be achieved by adding a "no-op" instruction with fields that identify the pipeline for execution of the instruction, or by supplying the information relating to the parallel instructions in another manner. It therefore should be appreciated that the manner in which the instruction and pipeline identifier arrives at the crossbar to be processed is somewhat arbitrary. I use the word "associated" herein to designate the concept that the pipeline identifiers are not required to have a fixed relationship to the instruction words. That is, the pipeline identifiers need not be embedded within the instructions themselves by the compiler. Instead they may arrive from another means, or on a different cycle.

[0029] Figure 2 is a simplified diagram illustrating the secondary cache, the predecoder, and the instruction cache. This figure, as well as Figures 3, 4 and 5, are used to explain the manner in which the instructions tagged with the pipeline identifier are routed to their designated instruction pipelines.

[0030] In Figure 2, for illustration, assume that groups of instructions to be executed in parallel are fetched in a single transfer across a 256 bit (32 byte) wide path from a secondary cache 50 into the predecoder 60. As explained above, the predecoder prefixes the pipeline "P" field to the instruction. After predecoding the resulting set of instructions is transferred into the primary instruction cache 70. At the same time, a tag is placed into the tag field 74 for that line.

[0031] In the preferred embodiment the instruction cache operates as a conventional physically-addressed instruction cache. In the example depicted in Figure 2, the instruction cache will contain 512 bit sets of instructions of eight instructions each, organized in two compartments of 256 lines.

[0032] Address sources for the instruction cache arrive at a multiplexer 80 that selects the next address to be fetched. Because preferably instructions are always machine words, the low order two address bits <1:0> of the 32 bit address field supplied to multiplexer 80 are discarded. These two bits designate byte and half-word boundaries. Of the remaining 30 bits, the next three low order address bits <4:2>, which designate a particular instruction word in the set, are sent directly via bus 81 to the associative crossbar. The next low eight address bits <12:5> are supplied over bus 82 to the instruction cache 70 where they are used to select one of the 256 lines in the instruction cache. Finally, the remaining 19 bits of the virtual address <31:13> are sent to the translation lookaside buffer (TLB) 90. The TLB translates these bits into the high 19 bits of the physical address. The TLB then supplies them over bus 84 to the instruction cache. In the cache they are compared with the tag of the selected line, to determine if there is a "hit" or a "miss" in the instruction cache.

[0033] If there is a hit in the instruction cache, indicating that the addressed instruction is present in the cache, then the selected set of instructions is transferred across the 512 bit wide bus 73 into the associative crossbar 100. The associative crossbar 100 then dispatches the addressed instructions to the appropriate pipelines over buses 110, 111,...,117. Preferably the bit lines from the memory cells storing the bits of the instruction are themselves coupled to the associative crossbar. This eliminates the need for numerous sense amplifiers, and allows the crossbar to operate on the lower voltage swing information from the cache line directly, without the normally intervening driver circuitry to slow system operation.

[0034] Figure 3 illustrates in more detail one embodiment of the associative crossbar. A 512 bit wide register 130, which represents the memory cells in a line of the cache (or can be a physically separate register), contains at least the set of instructions capable of being issued. For the purposes of illustration, register 130 is shown as containing up to eight instruction words W0 to W7. Using means described in the copending application referred to above, the instructions have been sorted into groups for parallel execution. For illustration here, assume the instructions in Group 1 are to be dispatched to pipelines 1, 2 and 3; the instructions in Group 2 to pipelines 1, 3 and 6; and the instructions in Group 3 to pipelines 1 and 6. The decoder select signal enables only the appropriate set of instructions to be executed in parallel, essentially allowing register 130 to contain more than just one set of instructions. Of course, by only using register 130 only for one set of parallel instructions at a time, the decoder select signal is not needed.

[0035] As shown in Figure 3, the crossbar switch itself consists of two sets of crossing pathways. In the horizontal direction are the pipeline pathways 180, 181,..., 187. In the vertical direction are the instruction word paths, 190, 191, ..., 197. Each of these pipeline and instruction pathways is themselves a bus for transferring the instruction word. Each horizontal pipeline pathway is coupled to a pipeline execution unit 200, 201, 202, ..., 207. Each of the vertical instruction word pathways 190, 191, ..., 197 is coupled to an appropriate portion of register or cache line 130.

[0036] The decoders 170, 171, ..., 177 associated with each instruction word pathway receive the 4 bit pipeline code from the instruction. Each decoder, for example decoder 170, provides eight 1 bit control lines as output. One of these control lines is associated with each pipeline pathway crossing of that instruction word pathway. Selection of a decoder as described with reference to Figure 3 activates the output bit control line corresponding to that input pipe number. This signals the crossbar to close the switch between the word path associated with that decoder and the pipe path selected by that bit line. Establishing the cross connection between these two pathways causes a selected instruction word to flow into the selected pipeline. For example, decoder 173 has received the pipeline bits for word W3. Word W3 has associated with it pipeline path 1. The pipeline path 1 bits are decoded to activate switch 213 to supply instruction word W3 to pipeline execution unit 201 over pipeline path 181. In a similar manner, the identification of pipeline path 3 for decoder D4 activates switch 234 to supply instruction word W4 to pipeline path 3. Finally, the identification of pipeline 6 for word W5 in decoder D5 activates switch 265 to transfer instruction word W5 to pipeline execution unit 206 over pipeline pathway 186. Thus, instructions W3, W4 and W5 are executed by pipes 201, 203 and 206, respectively. The pipeline processing units 200, 201, ..., 207 shown in Figure 3 can carry out desired operations. In a preferred embodiment of the invention, each of the eight pipelines first includes a sense amplifier to detect the state of the signals on the bit lines from the crossbar. In one embodiment the pipelines include first and second arithmetic logic units; first and second floating point units; first and second load units; a store unit and a control unit. The particular pipeline to which a given instruction word is dispatched will depend upon hardware constraints as well as data dependencies.

[0037] Figure 4 is a diagram illustrating another embodiment of the associative crossbar. In Figure 4 nine pipelines 0 - 8 are shown coupled to the crossbar. The decode select is used to enable a subset of the instructions in the register 130 for execution just as in the system of Figure 3.

[0038] The execution ports that connect to the pipelines specified by the pipeline identification bits of the enabled instructions are then selected to multiplex out the appropriate instructions from the contents of the register. If one or more of the pipelines is not ready to receive a new instruction, a set of hold latches at the output of the execution ports prevents any of the enabled instructions from issuing until the "busy" pipeline is free. Otherwise the instructions pass transparently through the hold latches into their respective pipelines. Accompanying the output of each port is a "port valid" signal that indicates whether the port has valid information to issue to the hold latch.

[0039] Figure 5 illustrates an alternate embodiment for the invention where pipeline tags are not included with the instruction, but are supplied separately, or where the cache line itself is used as the register for the crossbar. In these situations, the pipeline tags may be placed into a high speed separate cache memory 200. The output from this memory can then control the crossbar in the same manner as described in conjunction with Figure 3. This approach eliminates the need for sense amplifiers between the instruction cache and the crossbar. This enables the crossbar to switch very low voltage signals more quickly than higher level signals, and the need for hundreds of sense amplifiers is eliminated. To provide a higher level signal for control of the crossbar, sense amplifier 205 is placed between the pipeline tag cache 200 and the crossbar 100. Because the pipeline tag cache is a relatively small memory, however, it can operate more quickly than the instruction cache memory, and the tags therefore are available in time to control the crossbar despite the sense amplifier between the cache 200 and the crossbar 100. Once the switching occurs in the crossbar, then the signals are amplified by sense amplifiers 210 before being supplied to the various pipelines for execution.

[0040] The architecture described above provides many unique advantages to a system using this crossbar. The crossbar described is extremely flexible, enabling instructions to be executed sequentially or in parallel, depending entirely upon the "intelligence" of the compiler. Importantly, the associative crossbar relies upon the content of the message being decoded, not upon an external control circuit acting independently of the instructions being executed. In essence, the associative crossbar is self directed.

[0041] Another important advantage of this system is that it allows for more intelligent compilers. Two instructions which appear to a hardware decoder (such as in the prior art described above) to be dependent upon each other can be determined by the compiler not to be interdependent. For example, a hardware decoder would not permit two instructions R1 + R2 = R3 and R3 + R5 = R6 to be executed in parallel. A compiler, however, can be "intelligent" enough to determine that the second R3 is a previous value of R3, not the one calculated by R1 + R2, and therefore allow both instructions to issue at the same time. This allows the software to be more flexible and faster.

[0042] Although the foregoing has been a description of the preferred embodiment of the invention, it will be

apparent to those of skilled in the art the numerous modifications and variations may be made to the invention without departing from the scope as described herein. For example, arbitrary numbers of pipelines, arbitrary numbers of decoders, and different architectures may be employed, yet rely upon the system we have developed.

Claims

1. A method for operating a processor, comprising the steps of:

   storing in a memory a plurality of instructions, each instruction being one of a plurality of instruction types, the instructions encoded in frames, each frame including a plurality of instruction slots and template bits that specifies instruction group boundaries within the frame, with an instruction group comprising a set of statically contiguous instructions that are executed concurrently;
   using a crossbar switch means (100) coupled to a plurality of execution units (0, ..., 7) to issue instructions in the instruction group in parallel to execution units (0, ..., 7) from the plurality of execution units (0, ..., 7) in response to the template bits;
   wherein each execution unit (0, ..., 7) of the plurality of execution units (0, ..., 7) is one of a plurality of execution unit types; and
   wherein each instruction type is executed on one or more execution unit types.

2. The method of claim 1 further wherein using the crossbar switch means (100) further comprises using the crossbar switch means (100) to couple the instructions to appropriate execution unit types in response to the template bits.

3. The method of claims 1 or 2 wherein the instruction types include integer instructions and floating-point instructions.

4. The method of claim of one of the claims 1 to 3 wherein the instruction types include load instructions and store instructions.

5. The method of one of the claims 1 to 4 wherein the execution units include an arithmetic logic unit and a floating-point unit.

6. The method of one of the claims 1 to 5 wherein the template bits comprises 4 bits.

7. The method according to one of the claims 1 to 6, wherein a byte order of the instructions in the frame in the memory are in a little-endian format or in a big-endian format.

8. The method according to one of the claims 1 to 7, wherein an instruction in the frame with the lowest memory address precedes an instruction in the frame with the highest memory address.

9. The method according to one of the claims 1 to 8, wherein the frame comprise at least first, second, and third instruction slots.

10. The method according to one of the claims 1 to 9, wherein the template bits are at least partly determined at compile time.

11. A processor comprising:

    an instruction set including instructions which address registers, each instruction being one of a plurality of instruction types, the instructions encoded in frames, each frame including a plurality of instruction slots and template bits that specifies instruction group boundaries within the frame, with an instruction group comprising a set of statically contiguous instructions that are executed concurrently;

    a plurality of execution units (0, ..., 7), each execution unit (0, ..., 7) being one of a plurality of execution unit types, wherein each instruction type is executed on one or more execution unit types; and
    a crossbar switch means (100) coupled to the plurality of execution units (0, ..., 7), the crossbar switch means (100) configured to issue instructions in the instruction group in parallel to execution units (0, ..., 7) from the plurality of execution units (0, ..., 7) in response to the template bits.

12. The processor of claim 11 wherein the crossbar switch means (100) is also configured to couple the instruction slots to the execution unit types in response to the template bits.

13. The processor of claims 11 or 12 wherein the instruction types include integer instructions and floating-point instructions.

14. The processor of one of the claims 11 to 13 wherein the instruction types include load instructions and store instructions.

15. The processor of one of the claims 11 to 14 wherein the execution units include an arithmetic logic unit and a floating-point unit.

16. The processor of one of the claims 11 to 15 wherein the template bits comprises 4 bits.

17. The processor according to one of the claims 11 to 16, further comprising a memory that stores the frames, a byte order of the frames in the memory being in a little-endian format or in a big-endian format.

18. The processor according to one of the claims 11 to 17, wherein an instruction in the frame with the lowest memory address precedes an instruction in the frames with the highest memory address.

19. The processor according to one of the claims 11 to 18, wherein the frame comprise at least first, second, and third instruction slots.

20. The processor according to one of the claims 11 to 19, wherein the template bits are at least partly determined at compile time.

21. A cache memory comprising:

a frame of instructions, the frame including a plurality of instructions and template bits that specifies instruction group boundaries within the frame, each instruction being one of a plurality of instruction types, and an instruction group comprising a set of statically contiguous instructions that are executed concurrently;

wherein each instruction type is to be executed on an execution unit (0, ..., 7) from a plurality of execution units (0, ..., 7), each execution unit (0, ..., 7) being one of a plurality of execution unit types; and

wherein instructions in the instruction group are issued by crossbar switch means (100) in parallel to execution units (0, ..., 7) from the plurality of execution units (0,..., 7) in response to the template bits.

22. The cache memory of claim 21 wherein the instructions are also issued by the crossbar switch means (100) to appropriate execution unit types in response to the template bits.

23. The cache memory of claims 21 or 22 wherein the instruction types include integer instructions and floating-point instructions.

24. The cache memory of one of the claims 21 to 23 wherein the instruction types include load instructions and store instructions.

25. The cache memory of one of the claims 21 to 24 wherein the execution units include an arithmetic logic unit and a floating-point unit.

26. The cache memory of one of the claims 21 to 25 wherein the template bits comprises 4 bits.

27. The cache memory according to one of the claims 21 to 26, wherein a byte order of instructions in the frame of instructions are stored in a little-endian format or in a big-endian format.

28. The cache memory according to one of the claims 21 to 27, wherein an instruction in the frame with the lowest memory address precedes an instruction in the frame with the highest memory address.

29. The cache memory according to one of the claims 21 to 28, wherein the frame comprise at least first, second, and third instruction slots.

30. The cache memory according to one of the claims 21 to 29, wherein the template bits are at least partly determined at compile time.

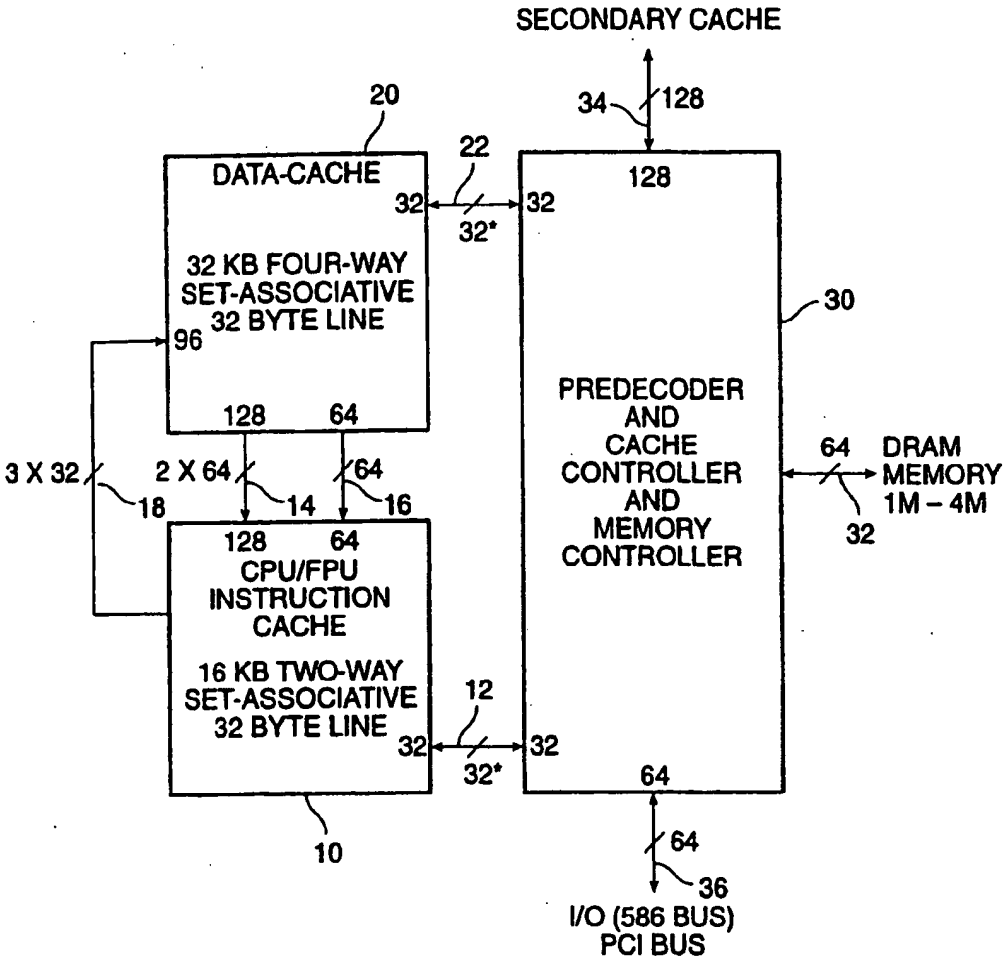
SECONDARY CACHE
34
128
20
DATA-CACHE
22
32
32*
32
128
32 KB FOUR-WAY
SET-ASSOCIATIVE
32 BYTE LINE
96
30
PREDECODER
AND
CACHE
CONTROLLER
AND
MEMORY
CONTROLLER
128
64
3 X 32
2 X 64
64
18
14
16
64
DRAM
MEMORY
1M – 4M
32
128
64
CPU/FPU
INSTRUCTION
CACHE
16 KB TWO-WAY
SET-ASSOCIATIVE
32 BYTE LINE
12
32
32*
32
64
10
64
36
I/O (586 BUS)
PCI BUS

FIG. 1

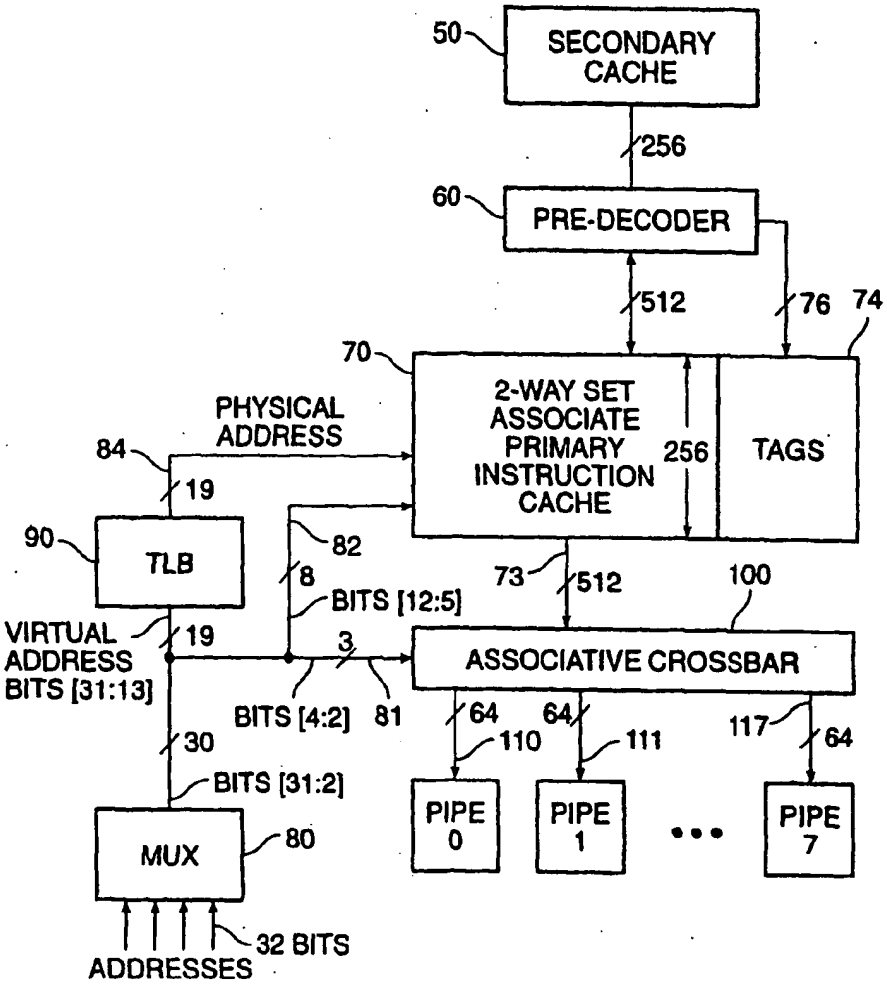
50
SECONDARY CACHE
256
60
PRE-DECODER
512
76
74
70
2-WAY SET ASSOCIATE PRIMARY INSTRUCTION CACHE
256
TAGS
PHYSICAL ADDRESS
84
19
90
TLB
82
8
BITS [12:5]
73
512
100
VIRTUAL ADDRESS BITS [31:13]
19
3
ASSOCIATIVE CROSSBAR
BITS [4:2]
81
64
64
117
64
110
111
30
BITS [31:2]
PIPE 0
PIPE 1
PIPE 7
MUX
80
32 BITS
ADDRESSES

FIG. 2

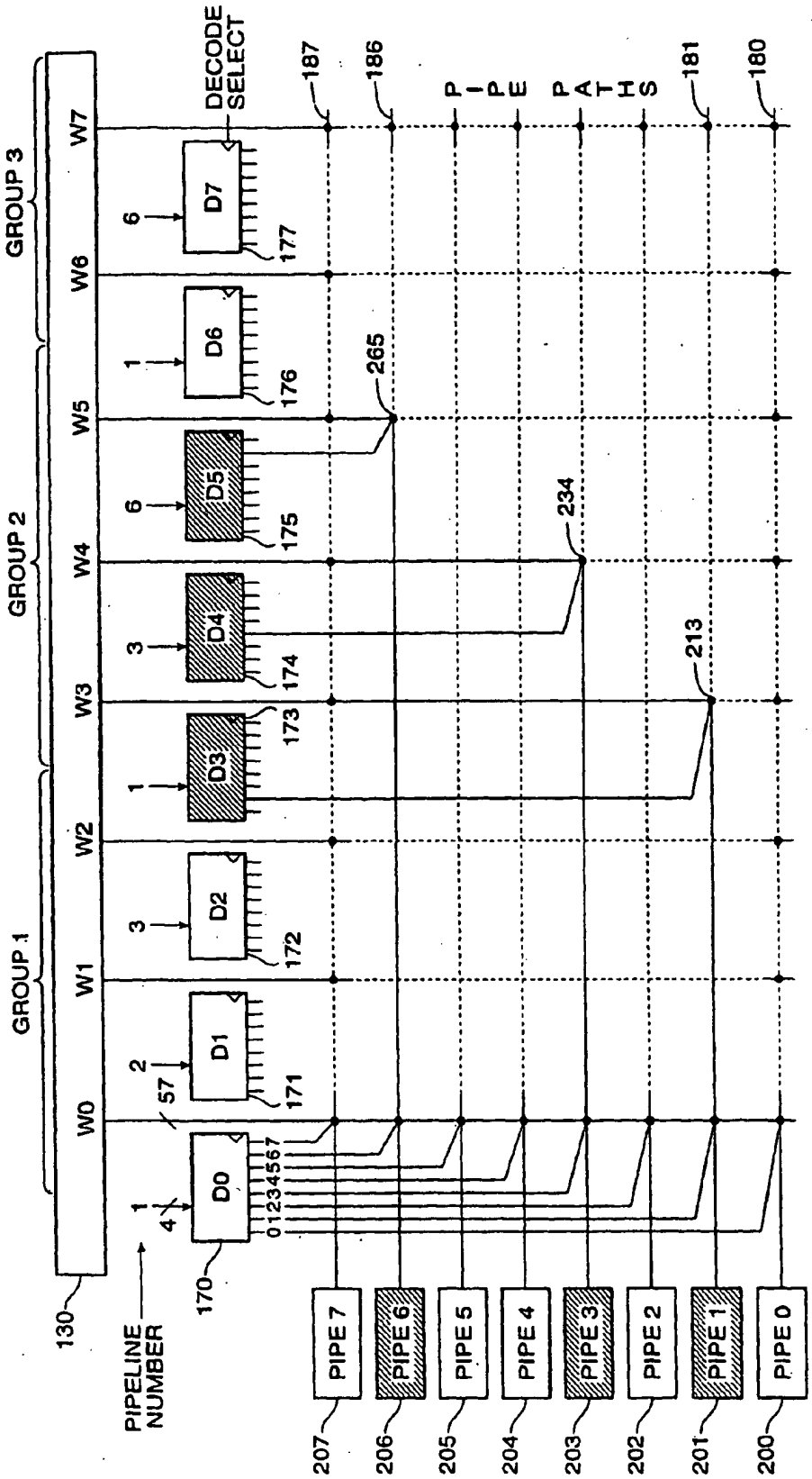
GROUP 1
GROUP 2
GROUP 3
W0
W1
W2
W3
W4
W5
W6
W7
D0
D1
D2
D3
D4
D5
D6
D7
DECODE SELECT
PIPELINE NUMBER
PIPE 7
PIPE 6
PIPE 5
PIPE 4
PIPE 3
PIPE 2
PIPE 1
PIPE 0
PIPE PATHS
130
170
171
172
173
174
175
176
177
180
181
186
187
200
201
202
203
204
205
206
207
213
234
265
57
0 1 2 3 4 5 6 7

FIG. 3

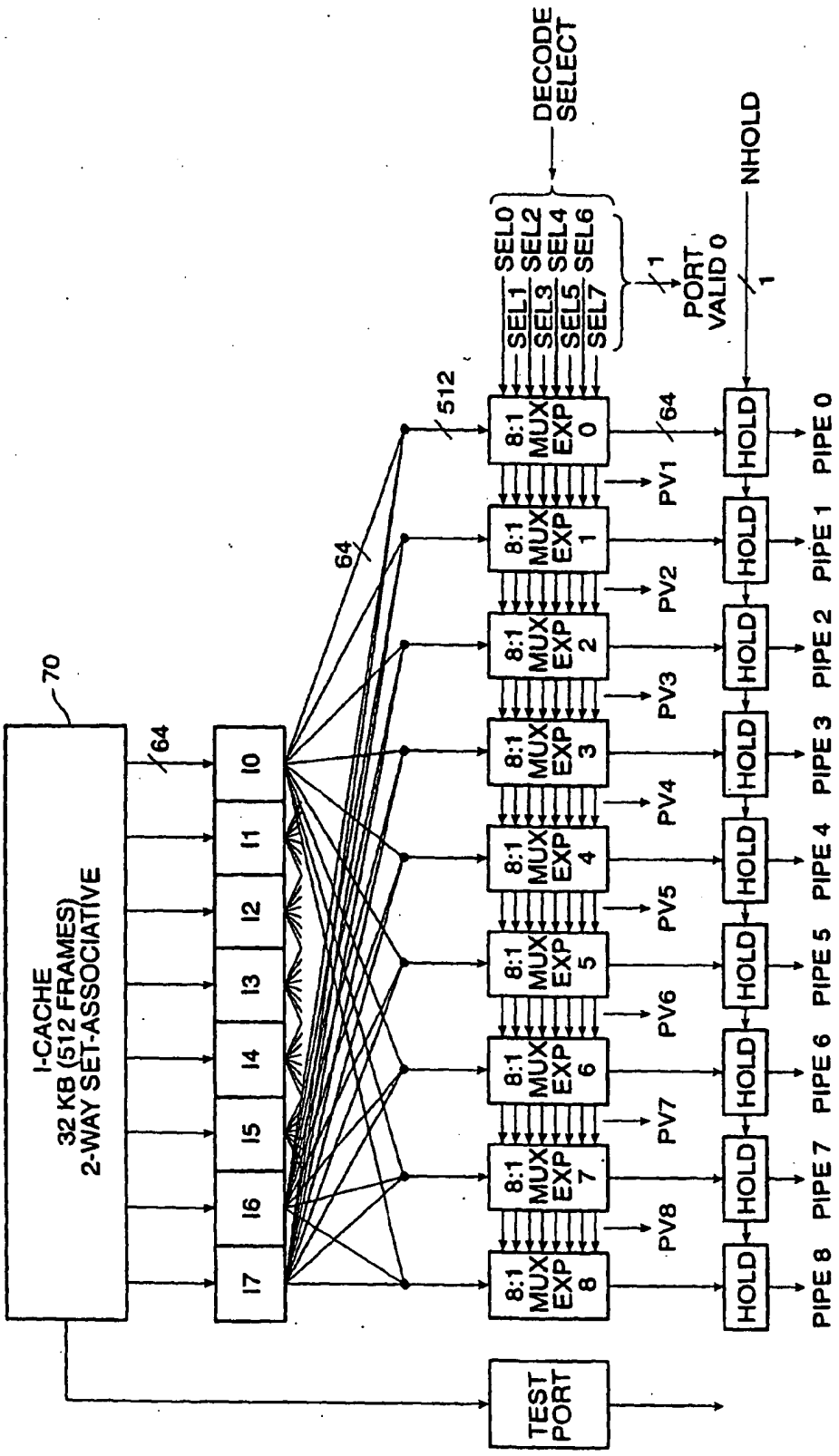
I-CACHE
32 KB (512 FRAMES)
2-WAY SET-ASSOCIATIVE
70
64
I0
I1
I2
I3
I4
I5
I6
I7
64
512
DECODE SELECT
SEL0
SEL1
SEL2
SEL3
SEL4
SEL5
SEL6
SEL7
1
PORT VALID 0
NHOLD
1
8:1 MUX EXP 0
8:1 MUX EXP 1
8:1 MUX EXP 2
8:1 MUX EXP 3
8:1 MUX EXP 4
8:1 MUX EXP 5
8:1 MUX EXP 6
8:1 MUX EXP 7
8:1 MUX EXP 8
64
PV1
PV2
PV3
PV4
PV5
PV6
PV7
PV8
HOLD
HOLD
HOLD
HOLD
HOLD
HOLD
HOLD
HOLD
HOLD
PIPE 0
PIPE 1
PIPE 2
PIPE 3
PIPE 4
PIPE 5
PIPE 6
PIPE 7
PIPE 8
TEST PORT

FIG. 4

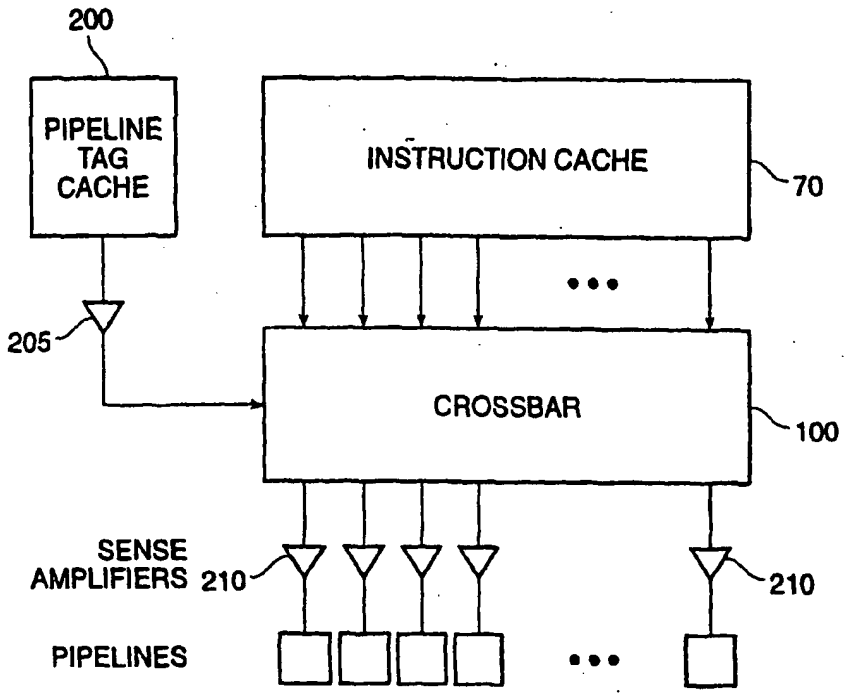
200
PIPELINE TAG CACHE
INSTRUCTION CACHE
70
205
CROSSBAR
100
SENSE AMPLIFIERS
210
210
PIPELINES

FIG. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 11 8582

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Y	DE GLORIA A ET AL: "A PROGRAMMABLE INSTRUCTION FORMAT EXTENSION TO VLIW ARCHITECTURES" PROCEEDINGS OF THE 6TH ANNUAL EUROPEAN CONFERENCE ON COMPUTER SYSTEMS AND SOFTWARE ENGINEERING, 4 May 1992 (1992-05-04), pages 35-40, XP000344165 ISBN: 0-8186-2760-3 * the whole document *	1-6, 8-16, 18-26, 28-30	G06F9/38
Y	EP 0 496 928 A (IBM) 5 August 1992 (1992-08-05) * the whole document *	1-6, 8-16, 18-26, 28-30	
A	EP 0 363 222 A (APOLLO COMPUTER) 11 April 1990 (1990-04-11) * summary of the invention *	1,11,21	
A	EP 0 449 661 A (KABUSHIKI KAISHA TOSHIBA) 2 October 1991 (1991-10-02) * the whole document *	1,11,21	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F
A	EP 0 426 393 A (FUJITSU LTD) 8 May 1991 (1991-05-08) * the whole document *	1,11,21	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	6 September 2001	Klocke, L

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 11 8582

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

06-09-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0496928 A	05-08-1992	US 5295249 A	15-03-1994
		AT 189540 T	15-02-2000
		AT 194236 T	15-07-2000
		BR 9101791 A	28-04-1992
		CA 2040637 A,C	05-11-1991
		DE 69131956 D	09-03-2000
		DE 69131956 T	20-07-2000
		DE 69132271 D	03-08-2000
		DE 69132271 T	21-12-2000
		EP 0481031 A	22-04-1992
		EP 0545927 A	16-06-1993
		EP 0825529 A	25-02-1998
		ES 2142304 T	16-04-2000
		HU 60048 A	28-07-1992
		HU 216990 B	28-10-1999
		JP 2046578 C	25-04-1996
		JP 6083623 A	25-03-1994
		JP 7078737 B	23-08-1995
		JP 6080489 B	12-10-1994
		JP 4505823 T	08-10-1992
		JP 6079273 B	05-10-1994
		JP 4506878 T	26-11-1992
		WO 9117496 A	14-11-1991
		WO 9117495 A	14-11-1991
		US 5465377 A	07-11-1995
		US 5448746 A	05-09-1995
		US 5446850 A	29-08-1995
		US 6029240 A	22-02-2000
		US 5701430 A	23-12-1997
		US 5504932 A	02-04-1996
		US 5303356 A	12-04-1994
EP 0363222 A	11-04-1990	US 5051885 A	24-09-1991
		DE 68927492 D	09-01-1997
		DE 68927492 T	03-04-1997
		JP 2224055 A	06-09-1990
		JP 2864421 B	03-03-1999
EP 0449661 A	02-10-1991	JP 2818249 B	30-10-1998
		JP 3282958 A	13-12-1991
		DE 69114333 D	14-12-1995
		DE 69114333 T	15-05-1996
		US 5377339 A	27-12-1994
EP 0426393 A	08-05-1991	JP 2835103 B	14-12-1998
		JP 3147021 A	24-06-1991

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 11 8582

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

06-09-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0426393 A		AU 625008 B	25-06-1992
		AU 6572290 A	01-08-1991
		CA 2029088 A,C	02-05-1991
		DE 69031899 D	12-02-1998
		DE 69031899 T	16-04-1998
		KR 9307041 B	26-07-1993
		US 5442762 A	15-08-1995

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82